一种优化计算 slater 投票获胜者的 picat 方法 *

敖 欢[1a, 1b]，王以松[1a, 1b†]，冯仁艳[1a, 1b]，邓周灰[2]，仝天乐[3]

(1. 贵州大学 a. 计算机科学与技术学院; b. 人工智能研究院，贵阳 550025; 2. 贵安科创产业发展有限公司 ,贵阳 550025; 3. 贵州黔驴科技有限公司 ,贵阳 550025)

摘 要：slater 投票规则是基于锦标赛的投票规则，主要是通过构造无环锦标赛，找到与原锦标赛差异最小的一个，从中选出获胜者。针对求解难度为 NP 难的 slater 投票算法，提出了一种基于相似候选项集的优化求解 slater 问题的 picat 方法。相比于非优化求解 slater 问题的方法，该方法缩小了 slater 算法的解空间，有效的减少了求解 slater 获胜者的计算量，提高了计算速度。实验结果表明，优化求解 slater 问题的 picat 方法的计算速度优于非优化的 picat 方法；当候选项人数少于 20 时，求解 slater 问题的回答集程序(ASP)方法的计算速度和计算能力优于优化的 picat 方法，但当候选项人数超过 30 时，优化的 picat 方法(用可满足问题求解器)的计算速度和计算能力优于 ASP 方法。

关键词：slater 投票问题；NP 难问题；约束满足问题；picat 程序设计；锦标赛；线性序列

中图分类号：TP3 doi: 10.19734/j.issn.1001-3695.2022.01.0013

Optimizing picat method for calculating slater voting winners

Ao Huan[1a, 1b], Wang Yisong[1a, 1b†], Feng Renyan[1a, 1b], Deng Zhouhui[2], Tong Tianle[3]

(1. *a. School of Computer Science & Technology*, *b. Institute of Artificial Intelligence*, *Guizhou University*, *Guiyang* 550025, *China*; 2. *Kechuang Industrial Development Company Limited*, *Guiyang* 550025, *China*; 3. *Guizhou Donkey Technologies Company Limited*, *Guiyang* 550025, *China*)

Abstract: The slater voting rule is a tournament-based voting rule. It mainly constructs a ringless tournament, finds the one with the smallest difference from the original tournament, and selects the winner from it. Aiming at the NP-hard slater voting algorithm, this paper proposed a picat method to solve the slater problem based on the optimization of similar candidate item sets. Compared with the non-optimized picat method for solving the slater problem, this method reduces the solution space of the slater algorithm, effectively reduces the amount of calculation for solving the slater winner, and improves the calculation speed. The experimental results show that the computational speed of the optimized picat method for solving the slater problem is better than that of the non-optimized; when the number of candidates is less than 20, the computational speed and computing power of the answer set program (ASP) method for solving the slater problem are better than those of the optimized picat method, but when the number of candidates exceeds 30, the optimized picat method (with a satisfiable problem solver) outperforms the ASP method in terms of computational speed and computational power.

Key words: slater voting; NP-hard problem; constraint satisfaction problem; picat programming; tournament; linear sequence

0 引言

可计算社会选择理论(Computational Social Choice, COMSOC)是社会选择理论和计算机科学融合而成的一个交叉学科，在人工智能、经济和计算性理论等领域有着广阔的应用前景[1~3]。社会选择研究各种社会决策能否尊重个体偏好和平衡利益分配，投票是最普遍的社会决策方式，其作用是将个人偏好转换为社会偏好,因此投票理论是可计算社会选择理论的主要研究内容之一，人们提出了各种投票投票算法规则：Kemeny[4]、Slater[5]、Banks[6]，其中 slater 投票算法的求解复杂度为 Θ_2^p－完全的[7]，该算法求解的 slater 问题是约束满足问题。

Picat 程序设计是一种描述性问题求解的新范例。Picat 程序设计针对约束满足问题的设计思想是：用一组规则描述所需要求解的问题，然后用相应的求解器计算出问题的解[8]。picat 融合了声明式语言特性和命令式语言特性，既有类似 Prolog 的声明式语言特性，又具有数组，函数等命令式语言的特性，因此用 picat 对组合优化问题进行建模非常方便[9]。文献[10~12]提出使用回答集程序设计(ASP)求解 slater 问题的方法。用 ASP 求解 slater 问题，虽然可以计算出至少一种解决方案，但当求解问题规模较大时，计算时间过长。鉴于在求解多人寻路问题(Multi-Agent Pathfinding Problem)的实验中，当问题规模较大时，picat 的求解时间少于 ASP 的[13]，此外，slater 问题是约束满足问题，而 picat 支持约束编程，内置了三种求解器，都可以用来求解约束满足问题，所以本文用 picat 求解 slater 问题。此外，文献[14]提出了优化求解 slater 问题的方法，本文会用 picat 实现优化的方法，以此求解 slater 问题。

本文介绍 slater 投票算法以及 picat，阐述用 picat 优化求解 slater 问题的方法，并分析其正确性。

1 背景知识

1.1 问题描述

投票活动中，当候选人数达到 3 个或 3 个以上，在成对

收稿日期：2022-01-13；**修回日期：**2022-03-01 **基金项目：**国家自然科学基金资助项目(61976065)；国家自然科学基金资助项目(U1836205)

作者简介：敖欢(1997-)，男，湖南衡阳人，硕士研究生，主要研究方向为 picat 程序设计；王以松(1975-)，男，(通信作者)，教授，博士，主要研究方向为知识表示与推理，机器学习(yswang@gzu.edu.cn)；冯仁艳(1990-)，贵州黔西人，博士研究生，主要研究方向为知识表示与推理；邓周灰(1978-)，男，贵州贵阳人，硕士研究生，主要研究方向为机器学习、统计分析、算法分析与优化；仝天乐(1990-)，男，贵州贵阳人，硕士研究生，主要研究方向为云计算、移动互联网、虚拟现实.

候选项的选择结果中可能会出现“投票悖论”问题，即循环的选择结果[15]。例如，假设有 3 个候选项 A，B，C，三个人的投票分别为：$A>B>C$，$B>C>A$ 和 $C>A>B$ ($x>y$ 表示 x 比 y 好)。按照少数服从多数原则，这样无法确定获胜者(或者 A、B、C 都可能是获胜者)。针对投票悖论问题，研究者们提出了各种投票算法。slater 投票算法就是其中之一。

1.2　slater 投票

slater 投票是基于锦标赛的一种投票方式。锦标赛 $T=(C,P)$，C 是候选项集，P 是集 C 上的完全二元关系，表示两两候选项之间的集体偏好关系，集体偏好关系反映的是两个候选项之间支持票数的多少。设 x 和 y 为任意两个候选项，$(x,y)\in P$ (记为 $x>y$)，表示支持 x 的票数多于支持 y 的。因此，锦标赛是个反对称的有向完全简单图。对于具有 n 个元素的集合 C 和 C 中的元素 $x_i(1\le i\le n)$，称 $l=x_1>\ldots>x_n(x_i\in C,1\le i\le n)$ 为集合 C 上的任一严格线性序列，也记 $l=\{(x_i,x_j)|1\le i<j\le n\}$。锦标赛可能不能形成一个严格的线性序列。

定义 1　slater 分数。给定锦标赛 $T=(C,P)$，C 上的一严格线性序列 l 关于 T 的 slater 分数定义为：$|\{(x,y)|(x,y)\in l 且 (y,x)\in P\}|$。

具有最小 slater 分数的任何 C 上的线性序列，被称为 slater 序列，slater 获胜者即 slater 序列的首项。slater 问题即求解锦标赛 T 的 slater 获胜者。

研究者们认为 slater 问题至少是 NP 难问题，对 slater 问题的计算复杂度的最新研究表明 slater 问题是 Θ_2^p 类问题[5,16]。Lampis 证明 slater 问题是 Θ_2^p -完全问题[7]。Slater 问题的研究目标是研究在多项式时间之内计算出任何锦标赛的 slater 序列的方法。该方法可以用于求解回馈弧集问题(Feedback Arc Set problem),该问题是锦标赛研究中的 NP 完全问题[14,17~19]。

在 2019 年，Bachmeier 指出即使只有 7 名选民的情况下，求解 slater 问题的计算复杂度仍是 NP 难的[20]。2021 年，Lampis 指出即使只有 7 名选民的情况下，求解 slater 问题的计算复杂度是 Θ_2^p -完全的[7]。因此 slater 问题是难计算的。

用 picat 求解 slater 问题的难点，首先是如何构造可以描述投票描述的谓词，根据投票描述的谓词构造锦标赛。2019 年，徐珩僭等人提出了一种基于回答集程序(ASP)的计算 slater 问题的方法[11,12]。本文借鉴了其中定义投票描述的谓词构造锦标赛。

再者，设计优化求解的算法是又一难点。构造完锦标赛之后，非优化求解 slater 问题的方法是利用求解器枚举所有的线性序列，与锦标赛比较，找出其中差异最小的，即 slater 分数最小的，从而求出获胜者。其时间复杂度是 O(n!)。这样随着问题规模的增大，用非优化方法求解 slater 获胜者的计算时间过长。

2006 年 Conitzer 提出优化求解 slater 问题的新方法，他为候选项引入相似性概念：相似的项与任何其他项具有相同的偏好关系。通过搜寻原锦标赛中的相似项集构造加权锦标赛，该加权锦标赛的每一个顶点是一个相似项集，先计算加权锦标赛的 slater 获胜者，该获胜者是一个相似项集，该相似项集中有一个候选项是原锦标赛的获胜者。然后递归求解由该相似项集构成的子锦标赛的 slater 获胜者，该 slater 获胜者即为原锦标赛的 slater 获胜者[14]。下面详细介绍其基本概念。

定义 2　极大相似项集。给定锦标赛 $T=(C,P)$,若 $S\subseteq C$，且对于任意两个候选项 $x,y\in S$，对于任意 $z\in C-S$，$x>z$ 当且仅当 $y>z$，或者 $z>x$ 当且仅当 $z>y$，则称 S 为极大相似项集，称 S 中的候选项为相似项。

极大相似项集分为两种，非平凡相似项集和平凡相似项集。单个候选项和全体候选项都能构成一个相似项集，即若 $|S|=1$ 或者 $|S|=|C|$ 则称 S 为平凡相似项集。若锦标赛中只存在平凡相似项集，则不能用优化方法求解 slater 获胜者。若 $1<|S|<|C|$，则称 S 为非平凡相似项集。优化算法的关键步骤是找出锦标赛中存在的非平凡相似项集。

定义 3　加权锦标赛。给定锦标赛 $T=(C,P)$，其加权锦标赛为 $T_w=(C_w,P_w)$，其中 $C_w=\{S_1,\ldots,S_k\}$ 是 C 的极大相似项集的集合，即每个 $S_i(1\le i\le k)$ 都是 C 上的一个极大相似集，其中 $k=|C_w|$，而且每个 S_i 都自带权重，权重值为 $|S_i|$。

给定加权锦标赛 $T_w=(C_w,P_w)$，C_w 上的一个严格线性序列 wl 的加权 slater 分数定义为

$$\sum_{(S_m,S_j)\in wl\Delta P_w}|S_m|\times|S_j| \tag{1}$$

加权锦标赛 T_w 的 slater 获胜者是具有最小加权 slater 分数的任何 C_w 上的线性序列的首项。

1.3　picat 程序设计

Picat(http://picat-lang.org/index.html)是通用目的编程语言，它结合了逻辑程序设计、函数式程序设计、约束程序设计和脚本语言等的特征；支持逻辑的合一、非确定选择、查表(tabling)等计算特征，也支持循环、数组、列表等控制和多种数据结构，基于高效的 prolog 引擎 B-Prolog；目前结合了可满足问题(SAT)求解器、约束问题(CP)求解器和混合整数规划(MIP)求解器，被广泛应用在组合优化、图搜索和大量逻辑难题[8,9]。

2　计算 slater 获胜者算法

2.1　构造相似项集

给定锦标赛 $T=(C,P)$,优化求解 slater 投票获胜者的第一步是找出锦标赛中存在的非平凡相似项集。

设 S 为 T 中的一个极大相似项集，因为寻找的是非平凡相似项集，因此 S 中至少有两个元素，因此任取两个不同的候选项 $x,y\in C$ 作为 S 的初始值，即 $S:=\{x,y\}$。如算法 1 中步骤 a)所示。根据定义 2 可知，对于任意的非平凡相似项集 S，满足如下性质 1：

不存在 $s_1,s_2\in S$，$c\in C-S$ 使得 $s_1>c>s_2$ (或者 $s_2>c>s_1$)，即不存在 $s_1,s_2\in S$，$c\in C-S$ 使得 $(s_1,c)\in P$ 并且 $(c,s_2)\in P$。性质 1 在后文 3 正确性分析中，将给出证明。

为了确保 x,y 满足性质 1，因此令 $S1:=\{c\in C|x>c>y\}$，$S2:=\{c\in C|y>c>x\}$，S1 和 S2 都是 S 的子集，如算法 1 中步骤 b)~d)所示。因为 $S1,S2\subseteq S$,所以对于任意 $c\in S1\cup S2$，也要满足性质 1，故若存在 $\{u,v\}\subseteq S$，$a\in C-S$，使得 $(u,a)\in P,(a,v)\in P$，则 $a\in S$。令 $S3=\{a\in C-S|\ u>a>v,\{u,v\}\subseteq S\}$，S3 也是 S 的子集。如算法 1 中步骤 e)-g)所示。

最后，输出相似项集 $S=\{x,y\}\cup S1\cup S2\cup S3$,如算法 1 中步骤 h)所示。

给定锦标赛 $T=(C,P)$ 及任选两个不同候选项 $x,y\in C$，计算包含 x,y 的非平凡相似项集的方法 MST(x,y,T)如下：

算法 1　MST($T=(C,P)$, $\{x,y\}\subseteq C$)

输入：$T=(C,P)$, $\{x,y\}\subseteq C$

输出：包含 x,y 的单个极大相似项集 S

a) $S:=\{x,y\}$;

b) $S1:=\{c\in C|x>c>y\}$;

c) $S2:=\{c\in C|y>c>x\}$;

d) $S:=S\cup S1\cup S2$;

e) $E:=C-S$;

f) while ($\exists a\in E,\{u,v\}\subseteq S$ 使得 $(u,a)\in P,(a,v)\in P$)

g) 　　$S:=S\cup\{a\};E:=E-\{a\}$

h) return S

2.2　构造加权锦标赛

给定锦标赛 $T=(C,P)$，优化求解 slater 投票获胜者的第二步是根据锦标赛 T 构造其加权锦标赛 $T_w=(C_w,P_w)$，主要是构造 C_w 和 P_w，且初始值都为空集，TL 表示可能存在新的相似项集的集合，其初始值为 C，Fail 表示不在同一个相似项集中的两个候选项的集合，其初始值为空值。如算法 2 中步骤 a)所示。

C_w 是锦标赛 T 中的所有极大相似项集的集合。换句话说，加权锦标赛中每个顶点都是一个极大相似项集，无论平凡的还是非平凡的。而利用算法 1 计算出 T 中的极大相似项集的条件是，任取的两个候选项 x,y 可能互为相似项，即 $(x,y)\notin Fail$ 并且 x,y 不能是别的非平凡相似项集中的候选项，即 $\{x,y\}\subseteq TL$，此外，TL 中至少要有两个候选项待选，即 $|TL|>2$，如算法 2 中步骤 b)所示。

算法 1 的计算出来的极大相似项集 S 分为两种，若是非平凡相似项集，则 S 是 C_w 中的元素，若是平凡相似项集，则说明 x,y 不是同一个相似项集中的元素，因此 $\{(x,y),(y,x)\}$ 是 $Fail$ 的子集，如算法 2 中步骤 b)~g)所示。

任取两个相似项集 $S_1,S_2\in C_w$，$c_i\in S_1,c_j\in S_2$，因为 S_1,S_2 由相似项构成，相似项在锦标赛 T 中具有相同的偏好关系，所以，下面两种情况必然可以满足一个：

a) 对于每一个 c_i，c_j，都存在 $c_i\to c_j$

b) 对于每一个 c_i，c_j，都存在 $c_j\to c_i$

因此在加权锦标赛 $T_w=(C_w,P_w)$ 中，若 $(c_i,c_j)\in P$，则 $(S_1,S_2)\in P_w$；反之，$(c_j,c_i)\in P$，则 $(S_2,S_1)\in P_w$ 如算法 2 中步骤 j)~l)所示。

给定锦标赛 $T=(C,P)$，计算其加权锦标赛的方法 MST(T)如下：

算法 2　MST($T=(C,P)$)

输入：锦标赛 T

输出：由 T 的所有极大相似项集的集合 C_w 构成的加权锦标赛 T_w

a)　$C_w:=\varnothing$；$TL:=C$；$Fail:=\varnothing$；$P_w:=\varnothing$

b)　while（$|TL|$ ≥ 2 并且 $\{x,y\}\subseteq TL$ 并且 $(x,y)\notin Fail$）

c)　　$S:=$ MST(x,y,TL);

d)　　if($1<|S|<|C|$) then

e)　　　$C_w:=C_w\cup\{S\}$　；$TL:=TL-S$；

f)　　else

g)　　　$Fail:=Fail\cup\{(x,y),(y,x)\}$

h)　foreach($a\in TL$)

i)　$C_w:=C_w\cup\{\{a\}\}$；

j)　foreach（$\{x,y\}\subseteq C_w$）

k)　　if（$\exists u\in x,v\in y$ 使得 $(u,v)\in P$）then

l)　　$P_w:=P_w\cup\{(x,y)\}$

m)　Return　$T_w=(C_w,P_w)$

2.3　计算出加权锦标赛的一个 slater 获胜者

给定锦标赛 $T=(C,P)$，优化求解 slater 投票获胜者的第三步是计算出加权锦标赛 $T_w=(C_w,P_w)$ 的一个 slater 获胜者。

给定一个加权锦标赛 $T_w=(C_w,P_w)$，wl 为任意 C_w 上的严格线性序列，C_w 上具有最小加权 slater 分数的严格线性序列为 T_w 的 slater 序列，其首项为 T_w 的一个 slater 获胜者。所以主要分为两步：一，描述 C_w 上的严格线性序列 wl；二，求解出 C_w 上具有最小加权 slater 分数的严格线性序列。

对具有 k 个元素的 C_w 和 C_w 中的元素 $S_i(1\le i\le k)$，称 wl=$S_1>\text{K}>S_k$ 为 C_w 上的任意严格线性序列，又记作 $wl=\{(S_i,S_j)|1\le i<j\le k\}$，即每个严格线性序列可以看做一个有序二元组集合。

任意两个 C_w 上的严格线性序列之间的区别在于各个 S_i 的排列顺序。因此描述 wl 的关键在于描述 wl 中各个 S_i 的排列顺序。

设 $c:=\{s_i\,|\,1\le i\le k\}$，其中 $k=|C_w|$，$s_i\in\{1,..,k\}$，s_i 的取值表示任意 wl 中 S_i 的排列位置，而且 s_i 的取值范围为 1 到 k。例如，$C_w=\{S_1,S_2,S_3\}$，$c=\{3,1,2\}$，可以表示 C_w 上的一个线性序列 $S_2>S_3>S_1$。

集合 c 还有一个必要的约束条件，即 c 中 k 个元素的取值各不相同，$all_different(c)$ 可以确保 c 中的各个元素各不相等。$all_different(FVar)$ 是 picat 内置的约束谓词，它的作用是确保集合 $FVar$ 中任意两个元素各不相等。对于任意 $V1,V2\in FVar$，编译之后，$all_different(FVar)$ 会生成不等约束，即 $V1\ne V2$。所以，对于任意 $s_i,s_j\in c$，编译之后 $all_different(c)$ 会生成 $s_i\ne s_j$ 的约束，以确保 c 中的元素各不相同。例如，$|c|=3$，实例化 c 时，$\{1,1,1\}$ 或者 $\{2,1,2\}$ 都不会是 c。

所以令 $C_w:=\{S_1,\text{K},S_k\}$，$c:=\{s_1,\text{K},s_k\}$，其中 $s_i\in\{1,\text{K},k\}$，$1\le i\le k$ 并且 c 满足 $all_different(c)$，则 c 可以描述 C_w 上任意严格线性序列。如算法 3 中所示。

约束 Minimize $\sum_{\substack{1\le i<j\le k, s_i<s_j\\(S_j,S_i)\in P_w}}|S_i|\times|S_j|$ 可以求解出 C_w 上具有最小加权 slater 分数的严格线性序列。如算法 3 中步骤 b)所示。后文在第 3 章正确性分析中证明。

计算出加权锦标赛 T_w 的一个 slater 序列之后，该 slater 序列的首项即加权锦标赛 T_w 的一个 slater 获胜者。如算法 3 中步骤 c)所示。

给定一个加权锦标赛 $T_w=(C_w,P_w)$，计算出加权锦标赛 T_w 的一个 slater 获胜者方法 slater_winner_w($T_w=(C_w,P_w)$)如下：

算法 3　slater_winner_w($T_w=(C_w,P_w)$)

输入：加权锦标赛 T_w

输出：T_w 的一个 slater 胜者 $S_{\min(c)}$

a) 令 $C_w:=\{S_1,\text{K},S_k\}$，$c:=\{s_1,\text{K},s_k\}$ /* s_i 的取值表示 S_i 在一个严格线性序列中的排序*/

b) Minimize $\sum_{\substack{1\le i<j\le k, s_i<s_j\\(S_j,S_i)\in P_w}}|S_i|\times|S_j|$

s.t. $1\le i\le k$，$s_i\in\{1,\text{K},k\}$，all_different(c)

/*all_different(c)表示 c 中的各个变量的取值不同*/

/*可用 SAT、CP、MIP 等求解器计算上述优化 Minimize 问题*/

c) Return $S_{\min(c)}$

注意，算法 slater_winner_w 也可用于计算非加权锦标赛 $T=(C,P)$ 的 slater 获胜者，此时 C 中每一个候选项 $c_i(1\le i\le n)$ 都可以看做一个平凡相似项集 $\{c_i\}$。下面采用 Conitzer 方法计算锦标赛的 slater 获胜者。即 C_w 中每个相似项集恰好都只有 1 个候选项(权重都为 1)时，$C_w=\{\{c_1\},...,\{c_n\}\}$，则该加权锦标赛 $T_w=(C_w,P_w)$ 就是一个锦标赛 $T=(C,P)$，其获胜者就是 slater 获胜者，故该方法 slater_winner_w 称为计算 slater 获胜者的非优化方法。

2.4　计算锦标赛 T 的 slater 算法

给定锦标赛 $T=(C,P)$，构造加权锦标赛 $T_w=(C_w,P_w)$，计算出加权锦标赛 T_w 的一个 slater 获胜者为 S_w。而 S_w 是一个极大相似项集，优化求解 slater 投票获胜者的前三步如算法 4 中步骤 a)~c)所示。

优化求解 slater 投票获胜者的第四步是递归求解由 H 构成的子锦标赛 $T_{sub}=(S_w,P|S_w)$ 的 slater 获胜者，其中 $(P|S_w)\subseteq P$，$(P|S_w)$ 表示 $\{(x,y)|\,(x,y)\in S_w{}^2\cap P\}$，即由极大相似项集 S_w 中每一个候选项之间相连的边构成的边集。子锦标赛 T_{sub} 的 slater 获胜者即锦标赛 $T=(C,P)$ 的 slater 获胜者。如算法 4 中步骤 d)~e)所示。

给定锦标赛 $T=(C,P)$，计算锦标赛 T 的 slater 获胜者的算法如下：

算法 4　slater_winner($T=(C,P)$)

输入：锦标赛 $T=(C,P)$

输出：锦标赛的一个 slater 获胜者

a) 计算 C 的极大相似项集的集合 $C_w:=\{S_1,\cdots,S_k\}$

b) 由 C_w 和 T 构造出加权锦标赛 $T_w=(C_w,P_w)$

c) 计算出加权锦标赛 T_w 的一个 slater 获胜者 S_w :=slater_winner_w(T_w)

d) 递归求解子锦标赛 $T_{sub}=(S_w,P\,|\,S_w)$ 的 slater 获胜者 w:=slater_winner_w($T_{sub}=(S_w,P\,|\,S_w)$)

e) 返回 w

3　正确性分析

算法 4 即优化求解 slater 问题方法。现在分析 4 个步骤的正确性，以此证明优化求解 slater 问题的方法的正确性。

定理 1　若 S 为相似项集，则存在一个原锦标赛 T 上的 slater 序列，S 中的候选项能够构成该 slater 序列中连续的一块子序列。(因此，不存在 $s_1,s_2\in S,c\in C-S$ 使得 $(s_1,c)\in P$ 并且 $(c,s_2)\in P$)

证明　设有两个 C 上的严格线性序列 $\succ_1,\succ_2$，S 中的候选项在 $\succ_1$ 上被分割成 m ($m>1$)块；而 S 中的候选项在 $\succ_2$ 上被分割成 m-1 块。下面将阐述如何将 $\succ_1$ 转换成 $\succ_2$，而且 $\succ_1$ 和 $\succ_2$ 的 slater 分数相同。

设 $\succ_1$ 由 3 块组成：$\{s_i^1\},\{s_i^2\}\subseteq S$ 和 $\{c_i\}\subseteq C-S$，S 中的候选项被 $\{c_i\}$ 分割成 2 块：$s_1^1 \succ_1 s_2^1 \succ_1 \cdots \succ_1 s_{l_1}^1 \succ_1 c_1 \succ_1 c_2 \succ_1 \cdots \succ_1 c_l \succ_1 s_1^2 \succ_1 s_2^2 \succ_1 \cdots \succ_1 s_{l_2}^2$。因为 S 是相似项集，则给定一个 c_i，其在锦标赛 T 中与每一个 s_i^j 具有相同的偏好关系，要么是 $c_i\to s_i^j$，要么是 $s_i^j\to c_i$。因此，下面两种情况，$\{c_i\}$ 必然能满足其中一种：

a) $\{c_i\}$ 中至少有一半 c_i 使得 $c_i\to s_i^j$。

b) $\{c_i\}$ 中至少有一半 c_i 使得 $s_i^j\to c_i$。

如果是情况 a)，则可以将 $\succ_1$ 转换 $c_1 \succ_2 c_2 \succ_2 \cdots \succ_2 c_l \succ_2 s_1^2 \succ_2 s_2^2 \succ_2 \cdots \succ_2 s_{l_2}^2 \succ_2 s_1^1 \succ_2 s_2^1 \succ_2 \cdots \succ_2 s_{l_1}^1$。设 $\{s_i^1\}$ 上的 slater 分数为 A1，$\{s_i^2\}$ 上的 slater 分数为 A2，因为 $l=|\{c_i\}|$，假设 $\{c_i\}$ 中有 p 个 c_i 使得 $c_i\to s_i^j$，q 个 c_i 使得 $s_i^j\to c_i$，而且 $p+q=l$。那么 $\succ_2$ 和 $\succ_1$ 的 slater 分数都是 $l+A1+A2$。同理，如果是情况 b) 则可以将 $\succ_1$ 转换为 $s_1^2 \succ_2 s_2^2 \succ_2 \cdots \succ_2 s_{l_2}^2 \succ_2 s_1^1 \succ_2 s_2^1 \succ_2 \cdots \succ_2 s_{l_1}^1 \succ_2 c_1 \succ_2 c_2 \succ_2 \cdots \succ_2 c_l$，$\succ_2$ 的 slater 分数也是 $l+A1+A2$。换句话说，$\succ_1$ 等价于 $\succ_2$。

上述转换在 S 被分割成 m ($m>2$)时，同样适用，重复多次这种转换，可以将原来的线性序列转换成新的线性序列，原来的线性序列中同一相似项集 S 中的元素被分割成多块，新的线性序列中同一相似项集 S 中的候选项是一块的。而且两个线性序列的 slater 分数是一样的。定理 1 得证。

算法 1 构造相似项集的方法是正确的，且每个相似项集都是极大的。分析如下：

设 $S\subseteq C$ 且 $|S|\geq 2$，根据投票悖论的定义可知，$|C|\geq 3$，下面分情况分析构造相似项集 S 的过程：

a) 当 $|S|=2,|C|\geq 3$ 时，$s_1,s_2\in S$，因为 $s_i(i=1,2)$ 是相似项，故对于任意 $t\in C-S$，每一个 s_i，存在 $t>s_i$ 或者 $s_i>t$，不存在 $s_1>t>s_2$。

b)当 $|S|=|C|=3$ 时，则 S 为平凡相似项集，$S=C$，因为 $t\in S$，故不存在 $s_1>t>s_2$。

c)当 $|S|=3,|C|>3$ 时，设有一严格序列 $\succ_3$ 中，C 被分成三块 $\{c_i^1\},\{c_j^2\},\{t\}$：$c_1^1 \succ_3 c_2^1 \succ_3 \cdots \succ_3 c_{l_3}^1 \succ_3 s_1 \succ_3 t \succ_3 s_2 \succ_3 c_1^2 \succ_3 c_2^2 \succ_3 \cdots \succ_3 c_{l_4}^2$，其中，对于每一个 c_i^j ,对于任意 $r\in\{s_1,s_2,t\}$，存在 $c_i^j>r$ 或者 $r>c_i^j$。根据极大相似项集的定义以及定理 1 可知，因为 $s_1,s_2\in S$，故 $t\in S$。因此不存在 $s_1>t>s_2$。

d)当 $|S|>3,|C|>3$ 时，根据情况 b)，c) 的分析，同理可知，对于任意 $s_1,s_2\in S$，$t\in C-S$，不存在 $s_1>t>s_2$，因为若 $s_1>t>s_2$，则 $t\in S$。

综上所述，在锦标赛 $T=(C,P)$ 中，任意 $s_1,s_2\in S$，$t\in C-S$，若 $s_1>t>s_2$，则 $t\in S$。因此算法 1 构造相似项集的方法是符合相似项集定义的，是正确的。

算法 2 中构造加权锦标赛的方法是正确的，分析如下：给定锦标赛 $T=(C,P)$，构造加权锦标赛 $T_w=(C_w,P_w)$，加权锦标赛中每个顶点都是一个相似项集，无论是非平凡相似项集还是平凡相似项集。T_w 中一个顶点(相似项集)与其他顶点之间的边的指向性取决于该相似项集中候选项的偏好关系。2.2 中算法 2 的算法描述和形式化说明，符合加权锦标赛的定义，故算法 2 正确。

算法 3 计算出加权锦标赛 $T_w=(C_w,P_w)$ 的一个 slater 获胜者的方法是正确的，分析如下：T_w 上的严格线性序列 wl 的 slater 分数为 $\sum_{(S_m,S_j)\in wl\Delta P_w}|S_m|\times|S_j|$，其中 $wl\Delta P_w=(wl\cup P_w)-(wl\cap P_w)$，$wl=\{(S_m,S_j)\,|\,1\leq m<j\leq k,k=|C_w|\}$，$wl\Delta P_w$ 表示 wl 和 P_w 中存在差异的边的集合，其中，若 $(S_m,S_j)\in wl\Delta P_w$，则 $(S_j,S_m)\in wl\Delta P_w$。根据 2.3 中算法 3 的描述和形式化说明可知 $\sum_{\substack{1\leq i<j\leq k,s_i<s_j\\(S_{s_j},S_{s_i})\in P_w}}|S_i|\times|S_j|$ 和 $\sum_{(S_m,S_j)\in wl\Delta P_w}|S_m|\times|S_j|$ 是等价的。所以算法 3 中的 Minimize 约束求出加权锦标赛上加权 slater 分数最小的严格线性序列，并返回其首项。符合加权锦标赛的 slater 获胜者的定义。故算法 3 正确。

算法 4 中第四步递归求解子锦标赛 $T_{sub}=(S_w,P\,|\,S_w)$ 的 slater 获胜者是正确的。分析如下：

给定锦标赛 $T=(C,P)$，构造出加权锦标赛 $T_w=(C_w,P_w)$，在找出 T 中所有的极大相似项集之后，当需要计算出一个 T 的 slater 序列时，根据定理 1，可以将每一个相似项集 $S_i(1\leq i\leq k,k=|C_w|)$ 看做一个超级候选项，而且其权重为 $|S_i|$。先计算出由超级候选项构成的 slater 序列，而因为一个 S_i 内部的 slater 序列和超级候选项构成的 slater 序列以及其他极大相似项集内部的 slater 序列无关，所以再递归求出每个 S_i 内部的候选项构成的 slater 序列。这样就可以计算出锦标赛 T 的一个 slater 序列。

综合上述，算法 4 计算 slater 获胜者是符合文献[11]中提出的求解 slater 问题的定义。故算法 4 是正确的。

4　实验与分析

4.1　举例说明

给定锦标赛 $T=(C,P)$ 如图 1 所示。根据算法 2 计算出 T 中的所有的极大相似项集：$S_1=\{a,b,d\}$、$S_2=\{c\}$ 和 $S_3=\{e,f\}$，因此 $C_w=\{S_1,S_2,S_3\}$，因为 $a\in S_1,c\in S_2,e\in S_3$，又因为 $(c,a),(c,e),(a,e)\in P$，所以，$P_w=\{(S_2,S_1),(S_2,S_3),(S_1,S_3)\}$。由此，构造出加权锦标赛 $T_w=(C_w,P_w)$，如图 2 所示。

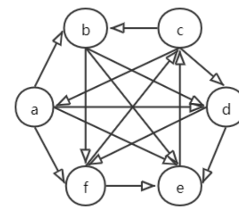


图 1　锦标赛

Fig. 1　Tournament

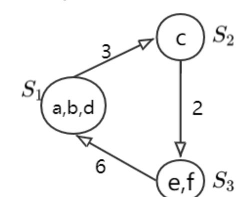


图 2　加权锦标赛

Fig. 2　Weighted tournament

根据算法 3 计算出 C_w 上加权 slater 分数最小的严格线性

系列，显然，$S_3 > S_1 > S_2$ 的加权 slater 分数为 2，最小，故 S_3 即加权锦标赛 T_w 的一个 slater 获胜者。

根据算法 4 的第四步，递归求解子锦标赛 $T_{sub}=(S_3, P|S_3)$，其中 $S_3=\{e,f\}$，$P|S_3=\{(e,f)\}$，则子锦标赛 T_{sub} 的 slater 获胜者为 e,因此，锦标赛 T 的一个 slater 胜者为 e。

4.2　实验与结果分析

本文实验是在 debian 9.2、257.288G 内存、Intel(R) Xeon(R) Gold 6132 CPU @ 2.60GHz 环境下进行，分别在不同候选项人数的情况下随机生成 100 组测试用例。每组测试用例其实是一些基础谓词，其中包括 votes(M,X,O,N),表示在第 N 种投票描述中，有 M 个选民支持 X 排在第 O 位；candidate(1..L),表示总共有 L 名候选项；voters(X)表示一共有 X 位选民参与投票。与文献[8]中的回答集程序进行对比实验,比较 picat 程序和 ASP 程序的平均计算时间，时间单位为秒(s)。所有实验代码及实验数据已上传至 github(https://github.com/Barnette-ao/picat)。

在第一组实验中，候选项数量分别为 5，10，15，20，单个测试用例的计算时间限制为 60 秒。因为 picat 内置了多种求解器：混合整数规划(MIP)，命题可满足性(SAT)，约束可满足性(CP)，本文分别用 ASP 方法、picat 三种求解器的优化和非优化方法来计算 slater 获胜者。优化方法即本文介绍的方法，给定锦标赛 $T=(C,P)$，构造加权锦标赛 $T_w=(C_w,P_w)$，先求解加权锦标赛的 slater 获胜者，然后再递归求解由加权锦标赛的 slater 获胜者构成的子锦标赛的 slater 获胜者，该 slater 获胜者即锦标赛的 slater 获胜者；非优化方法则是，给定锦标赛 $T=(C,P)$，求出与锦标赛差异最小的一个 C 上的严格线性序列，其首项即为锦标赛 T 的 slater 获胜者。

实验结果如表 1~2 所示，求解方法名的后缀为 0 表示优化方法 slater_winner(算法 4)，后缀为 1 表示非优化方法 slater_winner_w(算法 3)。表中“-”表示计算时间超出时间限制，结果表明：使用 picat 计算 slater 获胜者的优化方法比非优化的方法更有效(更少的平均时间和更少的超时实例数)；picat 的三种方法中，SAT 方法优于 CP 方法，CP 方法优于 MIP；另外，ASP(clingo5.2.2 https:// github.com/ potassco /clingo/releases)方法仍然是这些方法中最有效的。

表 1　各 picat 方法的平均计算时间对比

Tab. 1　Comparison of the average computing time of each picat method

求解方法	平均计算时间/s			
	C5	C10	C15	C20
Picat_SAT_0	0.0043	**1.36**	**8.48**	15.32
Picat_SAT_1	0.0090	7.81	25.58	57.89
Picat_CP_0	**0.0005**	3.77	15.24	**14.66**
Picat_CP_1	0.0011	55.87	-	-
Picat_MIP_0	0.0056	-	-	-
Picat_MIP_1	0.0110	-	-	-

第二组实验，进一步探索比较 ASP(clingo)和 picat 的 SAT 优化方法以求解更多的候选项 25、30、35、40、45 和 50，每个实例的求解时间限制为 600 秒，这两种方法的平均计算时间如图 3 所示，超出时间限制的实例的数量如图 4 所示。

由图 3 可知，当候选项个数等于或者大于 25 时，ASP 方法的平均计算时间多于 Picat 方法的平均计算时间。随着候选项个数的增多，ASP 方法的平均计算时间也越来越长。而随着候选项个数的增加，picat 方法的平均计算时间始终在 50 秒以下。由图 4 可知，当候选项的个数少于 20 时，ASP 能够计算出更多的实例，当候选项的个数多于 30 时，picat 能计算出更多实例。

表 2　各 picat 方法的超时实例个数对比

Tab. 2　Comparison of the number of timeout instances of each picat method

求解方法	超时实例个数(总数为 100)			
	C5	C10	C15	C20
Picat_SAT_0	**0**	**0**	**0**	**9**
Picat_SAT_1	0	0	0	10
Picat_CP_0	0	0	26	44
Picat_CP_1	0	0	100	100
Picat_MIP_0	0	100	100	100
Picat_MIP_1	0	100	100	100

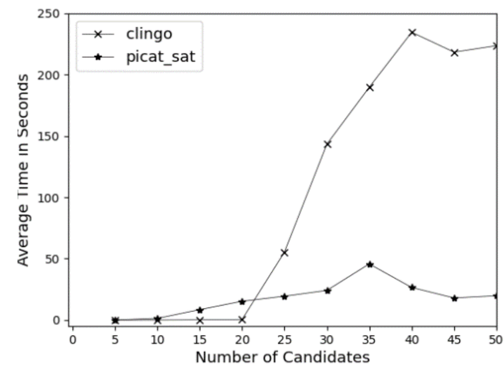


图 3　picat_sat 程序与 ASP 程序的平均计算时间对比

Fig. 3　Comparison of the average computing time of picat_sat program and ASP program

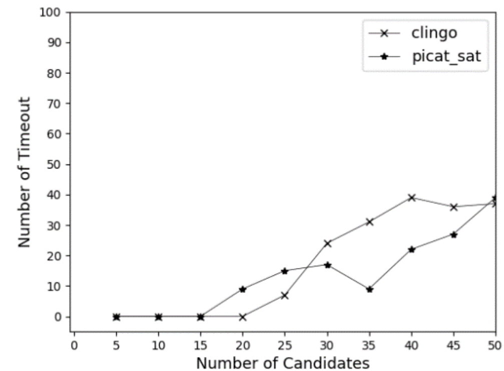


图 4　picat_sat 程序与 ASP 程序超时的实例数量对比

Fig. 4　Comparison of the number of instances of picat_sat_0 program and ASP program timeout

5　结束语

本文对 slater 投票问题进行了分析，以 picat 方法实现文献[14]中提出的优化求解 slater 问题的算法，并证明了 picat 方法的正确性。实验结果表明，当候选项少于 20 时，ASP 方法的计算效率和计算能力优于 picat 方法，当候选项多于 30 时，picat 的 SAT 方法计算效率和计算能力优于 ASP 方法。但本文方法的时间复杂度仍然较大，下一步将对其优化并计算其他投票策略的获胜者。

参考文献：

[1] Mattei N. Closing the loop: Bringing humans into empirical computational social choice and preference reasoning [C]// Proc of the 29th International Conference on International Joint Conferences on Artificial Intelligence (IJCAI 2021) . San Francisco: Margan Kaufmann, 2021: 5169-5173.

[2] Brandt F, Conitzer V, Endriss U, *et al.* Handbook of computational social choice [M]. Cambridge: Cambridge University Press, 2016, 1-20.

[3] Nardi O, Boixel A, Endriss U. A Graph-Based Algorithm for the Automated Justification of Collective Decisions [D] Amsterdam: University of Amsterdam, Institute for Logic, Language and Computation (ILLC) , 2021.

[4] Hamm T, Lackner M, Rapberger A. Computing Kemeny Rankings from d-Euclidean Preferences [C]// Proc of the 7th International Conference on Algorithmic Decision Theory. Berlin: Springer, 2021: 147-161.

[5] Boussaïri A, Chaïchaâ A, Chergui B, *et al.* Spectral Slater index of tournaments [J]. The Electronic Journal of Linear Algebra, 2022 (38): 170-178.

[6] Brill M, Schmidt-Kraepelin U, Suksompong W. Margin of victory for tournament solutions [J]. Artificial Intelligence, 2022 (302): 103600.

[7] Lampis M. Determining a Slater Winner is Complete for Parallel Access to NP [EB/OL]. (2021-04-07) [2022-02-27]. https://arxiv. org/pdf/2103. 16416. pdf

[8] Zhou Nengfa, Håkan K, Jonathan F. Constraint solving and planning with picat [M]. Berlin: Springer, 2015: 1-33.

[9] Zhou Nengfa. Modeling and Solving Graph Synthesis Problems Using SAT-Encoded Reachability Constraints in picat [C]// Proc of the 37th International Conference on Logic Programming (ICLP) , 2021: 165-178.

[10] De Haan R, Slavkovik M. Answer set programming for judgment aggregation [C]// Proc of the 28th International Joint Conference on Artificial Intelligence (IJCAI 2019) . Palo Alto, CA: AAAI Press, 2019: 1668-1674.

[11] 徐珩僭，王以松，冯仁艳. 一种用于 slater 与 Kemeny 投票求解的 ASP 方法 [J]. 计算机工程, 2019, 45 (09): 198-203. (Xu Hengjian. Wang Yisong, Feng Renyan. An A-SP method for calculating slater and Kemeny voting [J]. Computer Engineering, 2019, 45 (9): 198-203.)

[12] 徐珩僭. 计算社会选择选举问题及其变形的描述性求解 [D]. 贵州: 贵州大学, 2019. (XuHengjian. An Descriptive solution of computational social choice ele-ction problem and its variants [D]. Guizhou: Guizhou University, 2019.)

[13] Barták R. , Zhou Nengfa, Stern R, *et al.* Modeling and solving the multi-agent pathfinding problem in picat [C]// Proc of the 29th International Conference on Tools with Artificial Intelligence (ICTAI) . Piscataway, NJ: IEEE Press, 2017: 959-966.

[14] Conitzer V. Computing slater rankings using similarities among candidates [C]// The 21st Internation-al Conference on Artificial Intelligence and Soft Computing (ICAISC) . Palo Alto, CA: AAAI Press, 2006: 613-619.

[15] Nurmi H, Kacprzyk J, Zadrożny S. Collective Decisions: Theory, Algorithms And Decision Support Systems [M]. Berlin: Springer, 2022: 3-16.

[16] Govc D, Levi R, Smith J P. Complexes of tournaments, directionality filtrations and persistent homology [J]. Journal of applied and computational topology, 2021, 5 (2): 313-337.

[17] Lemus J, Marshall G. Dynamic tournament design: Evidence from prediction contests [J]. Journal of Political Economy, 2021, 129 (2): 383-420.

[18] Beretta L, Nardini F M, Trani R, *et al.* An Optimal Algorithm to Find Champions of Tournament Graphs [C]// Proc of the 26th International Symposium on String Processing and Information Retrieval. Berlin: Springer, 2019: 267-273.

[19] Baharev A, Schichl H, Neumaier A, *et al.* An exact method for the minimum feedback arc set problem [J]. Journal of Experimental Algorithmics (JEA) , 2021 (26): 1-28.

[20] Bachmeier G, Brandt F, Geist C, et al. k-Majority Digraphs and the Hardness of Voting with a Constant Number of Voters[J]. Journal of Computer and System Sciences, 2019(105): 130-157.